Second-Order Accurate Volume-of-Fluid Algorithms for Tracking Material Interfaces

James Edward Pilliod, Jr.
Center for Computational Sciences and Engineering
Computing Sciences Directorate
Lawrence Berkeley National Laboratory
1 Cyclotron Road
Berkeley, CA, 94720

Elbridge Gerry Puckett
Department of Mathematics
University of California
Davis, California 95616, USA

AMS Subject Classifications: 65M06, 76M20

Keywords: Advection Algorithms, Interface Tracking Algorithms, Volume-of-Fluid, Volume Tracking

Submitted to the Journal of Computational Physics on June 12, 1998.

Proposed Running Head: "Second-Order Accurate Volume-of-Fluid Algorithms"

Mailing Address:

James Edward Pilliod, Jr.

Center for Computational Sciences and Engineering

Lawrence Berkeley National Laboratory

Mail Stop 50D-3427

1 Cyclotron Road

Berkeley, CA, 94720

44 pages, 16 figures and 20 tables

Second-Order Accurate Volume-of-Fluid Algorithms for Tracking Material Interfaces†

James Edward Pilliod, Jr.
Lawrence Berkeley National Laboratory
1 Cyclotron Road
Berkeley, CA, 94720

Elbridge Gerry Puckett
Department of Mathematics
University of California
Davis, California 95616, USA

Abstract

We introduce two new volume-of-fluid interface reconstruction algorithms and compare the accuracy of these algorithms to four other widely used volume-of-fluid interface reconstruction algorithms. We find that the new methods are second-order accurate and the other algorithms are first-order accurate. We conjecture that a necessary and sufficient condition for a stable volume-of-fluid algorithm to be second-order accurate is that it reproduce straight lines (or planes in 3D) exactly. We also introduce a second-order, unsplit, volume-of-fluid advection algorithm that is based on a second-order, finite difference method for scalar conservation laws due to Bell, Dawson and Shubin. We test this advection algorithm by modeling several different interface shapes propagating in two simple incompressible flows and compare the results with the standard second-order, operator-split advection algorithm. Although both methods are second-order accurate, we find that the unsplit algorithm exhibits noticeably better resolution in regions where the interface has discontinuous derivatives, such as at corners.

† Work of both authors at U. C. Davis supported by the National Science Foundation under grants DMS-9104472 and DMS-9404410 and by the Mathematical, Information and Computational Sciences Division in the U. S. Department of Energy's Office of Energy Research under contract DE-FG03-95ER25271. Work of the second author at LBNL was provided by the Applied Mathematical Sciences Program of the DOE Office of Mathematics, Information, and Computational Sciences under contract DE-AC03-76SF00098 and by the Defense Nuclear Agency under IACRO 96-3075.

1. Introduction

There are numerous instances in which it is necessary to reconstruct or track the boundary between two materials in a numerical computation. Examples include numerical models of fluid jetting devices [14, 57, 63], weld pools [7], molten metal [31, 60], semiconductor device etching [1, 17, 18, 19] and thin flame models of combustion [8, 15, 53, 43]. An overview of the state of the field in the early 1980's may be found in [6]. However a number of new ideas have appeared since then, notably the level set approach of Osher and Sethian [1, 40, 54, 56, 67]. During the last decade there has also been considerable work devoted to developing algorithms that approximate the front as a collection of line segments (2D) or polygons (3D) (*e.g.*, [16, 61]) and on boundary integral methods (*e.g.*, [39, 55]).

In this article we study a class of interface tracking algorithms known as *volume-of-fluid* methods. In a volume-of-fluid method the motion of the interface itself is not tracked, but rather the volume of each material in each cell is evolved in time and the interface at the new time is reconstructed from the values of the volumes at this new time. For this reason volume-of-fluid methods are sometimes referred to as *volume tracking* methods (e.g., see [52]).

The basic idea behind a volume-of-fluid method is as follows.[1] Suppose that we wish to track the interface between two materials, say a dark fluid and a light fluid, in two dimensions. We begin by covering the problem domain with a grid with spacing $h = \Delta x = \Delta y$. With each grid cell we associate a number $f_{i,j}$ that represents the amount of dark fluid in the i,jth cell,

$$f_{i,j}\, h^2 \;=\; \text{volume of dark fluid in the}\, i,j\,\text{th cell.} \tag{1.1}$$

The number $f_{i,j}$ is called the *volume fraction* (of dark fluid) in the i,jth cell. It is apparent that

$$0 \le f_{i,j} \le 1\,, \tag{1.2}$$

that the volume fraction associated with the light fluid is $1 - f_{i,j}$ and that a portion of the interface lies in the i,jth cell if and only if $0 < f_{i,j} < 1$. The discrete variable $f_{i,j}$ is a discretization of the characteristic function associated with the dark fluid,

$$f(x,y) \equiv \begin{cases} 1 & \text{if there is dark fluid at the point } (x,y), \\ 0 & \text{if there is light fluid at the point } (x,y), \end{cases} \tag{1.3}$$

in the sense that

$$f_{i,j}\, h^2 \approx \int\int_{i,j\,\text{th cell}} f(x,y)\, dx\, dy\,. \tag{1.4}$$

[1] Here and in the remainder of this article we restrict the discussion to uniform square grids and two space dimensions. Neither of these restrictions are necessary. We employ them merely for simplicity and clarity of exposition.

Since the fluid type does not change along particle paths in an incompressible, non-reacting flow, the characteristic function f is passively advected with the flow. Hence, f satisfies the advection equation,

$$f_t + uf_x + vf_y = 0\,, \tag{1.5}$$

where $\mathbf{u} = (u, v)$ denotes the fluid velocity. If the flow is incompressible, then $\mathbf{u}$ satisfies

$$u_x + v_y = 0\,. \tag{1.6}$$

Multiplying (1.6) by f and adding it to (1.5) we obtain a conservation law for the characteristic function f,

$$f_t + (uf)_x + (vf)_y = 0\,. \tag{1.7}$$

Equation (1.7) reflects the fact that in an incompressible flow conservation of mass is equivalent to conservation of volume, and hence conservation of f.

In a compressible flow the velocity field $\mathbf{u}$ does not satisfy (1.6) and hence f is not conserved. However the mass of each material is conserved and therefore it is important that a numerical method for modeling this phenomena also conserve the mass of each fluid. It is relatively easy to design a volume-of-fluid interface tracking algorithm that does this (e.g., see [10, 36, 51]). Volume-of-fluid algorithms are the basis for most of the large application codes that are used at the national laboratories to model multi-phase, compressible phenomena on Eulerian grids (e.g., [2, 24, 25, 33]). These codes are also used extensively by geophysicists to model meteor impacts and related problems (e.g., see Melosh [34]).

Recently, there have been several important improvements to the basic volume-of-fluid methodology for modeling compressible flows. Colella, Glaz and Ferguson [10] have developed a model of interface motion in compressible flow in which (1.7) is modified by the addition of a term that accounts for the effect of isentropic volume changes due to changes in the pressure; i.e., changes in the specific volume V of the form $(\partial V/\partial P)_S$. Their method allows one to model disparities in the compressibility of two materials (*e.g.*, air and water) on a sub-grid scale. Puckett and Saltzman [51] have developed an algorithm for tracking gas interfaces in three space dimensions that is based on these ideas while Miller and Puckett [36] have developed a similar model for tracking the interface between two solids at very high pressures and temperatures (*e.g.*, magmas) in the hydrostatic limit (*i.e.*, without strength).

In this article we restrict ourselves to consideration of incompressible flow problems. One might expect the incompressible advection problem to be less difficult than the corresponding problem in compressible flow. However our experience has been that this is generally not true. The difficulty in modeling incompressible flow arises because f is also constrained by the maximum principle (1.2) but numerical errors in estimating the fluxes in (1.7) lead to overshoot and undershoot in the values of f. In practice we have found that for the

simple advection problems considered here these errors are on the order of machine zero (e.g., see Tables 4.9 and 5.6 below). For more difficult problems they tend to be on the order of one hundredth of a percent (e.g., see the computations in [47]).

Since in an incompressible flow f satisfies (1.7), the time update of the discrete variable $f_{i,j}$ can be accomplished with a conservative finite difference method. One can therefore draw on the vast body of knowledge for high-resolution numerical methods for conservation laws (e.g., [5, 9, 30, 62]) to devise a method for updating f numerically. In this article we present a volume-of-fluid advection algorithm that is based on ideas developed by Bell, Dawson and Shubin [5] to construct a finite difference method for modeling solutions of scalar conservation laws.

Volume-of-fluid methods have been in use for several decades. In one of the earliest implementations of these methods DeBar [13] used a volume-of-fluid algorithm in a two-dimensional Eulerian method to model compressible multi-phase flow. Another early algorithm of this type is the SLIC (Simple Line Interface Calculation) method of Noh and Woodward [38]. SLIC and its variants have been very widely used. For example, Colella, Henderson and Puckett used it to model shock wave refraction at a gas interface [11, 20, 45]. In [8] Chorin developed an improved version of SLIC in order to model flame propagation and combustion. Ghoniem, Chorin, and Oppenheim [15] and Sethian [53] used Chorin's version to model turbulent combustion, while Whitaker [64] used it to model Hele-Shaw flow. Another well-known volume-of-fluid algorithm is the VOF algorithm of Hirt and Nichols [23].[2] Several codes based on the VOF algorithm, namely SOLA-VOF [23, 37] and its descendants NASA-VOF2D [58], NASA-VOF3D [59], RIPPLE [26, 27] and FLOW3D [22] have been, and continue to be, widely used by researchers to model interfaces and free surfaces in industrial applications. For example, researchers at Xerox have used a modified version of these codes to model the flow in thermal ink jet devices [14, 57] and they have been used extensively by material scientists to model weld pools [7] and solidifying droplets [31, 60]. However, the volume-of-fluid algorithms in all of the methods just referred to are built around relatively crude interface reconstruction algorithms that rely on a piecewise constant or "staircase" representation of the interface, such as the one shown in Fig. 1.1c, and advection algorithms that are at best first-order accurate. More modern volume-of-fluid interface reconstruction methods use a linear approximation to the interface in each multifluid cell (e.g., [4, 28, 41, 46, 51, 52, 65]). This results in a piecewise-linear approximation to the interface as shown in Fig. 1.1d.

However, as we demonstrate in §3 below, a piecewise-linear approximation to the interface in each cell is not sufficient to guarantee a second-order accurate approximation to the interface. We demonstrate (numerically) that a sufficient condition for a volume-of-fluid interface reconstruction algorithm to be second-order

[2] Many workers use the acronym "VOF" - which stands for "Volume-of-Fluid" - to refer generically to any volume-of-fluid algorithm. However, we refrain from doing so since others use it to refer specifically to Hirt and Nichols' algorithm and the associated fluid dynamics code SOLA-VOF [23, 37].

accurate on smooth interfaces, is for the algorithm to reproduce linear interfaces exactly. In §3 we show that two interface reconstruction algorithms introduced by the authors (the Least Squares Volume-of-Fluid Interface Reconstruction Algorithm (LVIRA) [46] and the Efficient Least Squares Volume-of-Fluid Interface Reconstruction Algorithm (ELVIRA) [42]) have this property. These second-order accurate piecewise-linear interface reconstruction algorithms have been used extensively to model a variety of compressible and incompressible flows, including Richtmyer-Meshkov instability [32], shock refraction in gases [21, 48, 49] and shock refraction and impact jetting in solids [35, 36, 50] and the motion of fluid interfaces in variable density incompressible flow [47].

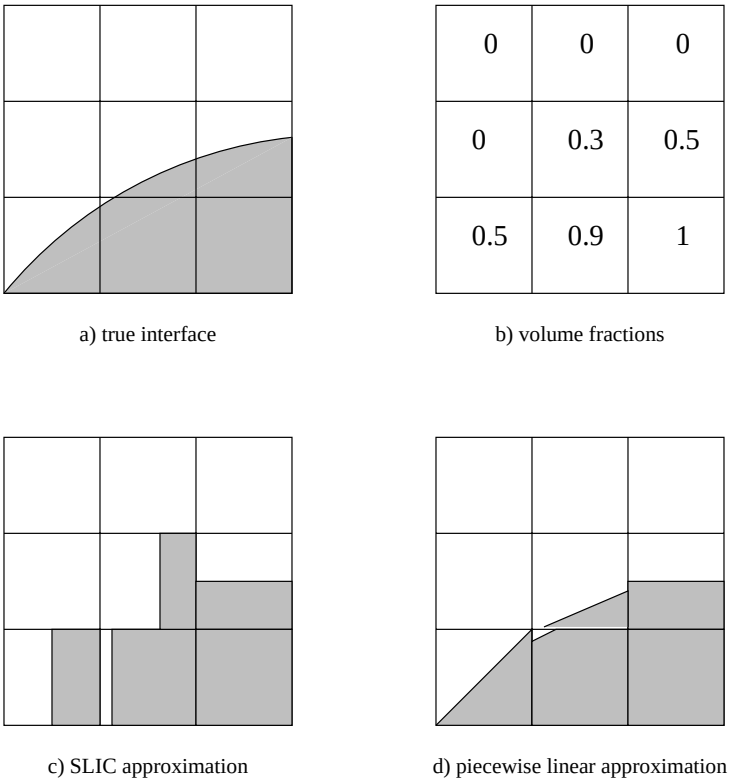


Figure 1.1 Volume-of-Fluid methods represent an interface (a) by storing volume fractions associated with the interface as shown in (b). An approximation to the interface is produced using an interface reconstruction method such as SLIC, shown in (c), or a more general piecewise linear approximation as in (d).

There seems to be a widely held belief in the CFD community that one cannot obtain high-order accuracy with a volume-of-fluid algorithm (e.g., see p. 26 of [61]). Perhaps this is due to the widespread use of SLIC and VOF, which are at best first-order accurate and can easily fragment a smooth front (e.g., see Figs. 4.2 and 4.3 below and Figs. 6 and 8 of [29]). One of the goals of this article is to demonstrate that one can construct high-order accurate volume-of-fluid interface tracking algorithms that are as effective, and for some problems more effective, than competing methods. There are four principal reasons for the effectiveness of

volume-of-fluid algorithms:

1) Volume-of-fluid algorithms naturally conserve the mass of each fluid. For incompressible flow this is because the advection algorithm is a conservative discretization of the conservation law (1.7) for f, which is equivalent to the mass conservation equation. In a compressible flow the mass of each fluid component must still be conserved even though the characteristic function f is not. In a volume-of-fluid method this can be easily arranged by appending a separate conservation law for the mass of each fluid to the original system of conservations laws (e.g., see [10, 36, 51]).

2) In both compressible and incompressible flows it is desirable, if not essential, that the location of the interface as determined by the interface tracking algorithm coincide with the location of the jump in density (and possibly other quantities) as determined by the underlying discretization of the fluid flow equations. Since the flux of a conserved quantity can be written in terms of the fluid volume that crosses a cell edge, it is a simple matter to enforce these constraints in a volume-of-fluid method.

3) Volume-of-fluid methods automatically handle changes in the global topology of the front, such as fronts that break up into droplets or fronts that collide with themselves and merge. This eliminates the algorithmic complexity that can occur when the front is modeled by a collection of line segments or polygons. Furthermore, the logical structure of the algorithm is not significantly more complicated in three dimensions than in two. This is in contrast to polygonal representations of the front in which the logical complexity increases substantially in going from two to three dimensions. (A discussion of the complexity issue can be found in [17].)

4) The work required to update the front location is entirely local; typically one needs the velocity and volume fractions in a 3×3 block (or $5 \times 5 \times 5$ block in 3D) of cells to update the volume fraction in the center cell. Since the interface is a codimension 1 set, the computational work required to update the location of the interface is typically $O(N^{d-1})$ for a problem on a grid with N^d cells in $d \geq 2$ space dimensions. Thus, the work required to update the front location is small compared to the work required to update the underlying velocity field. The local nature of volume-of-fluid algorithms also makes them amenable to efficient parallelization strategies.

In conclusion, volume-of-fluid methods can be naturally formulated in conservative finite difference form, thereby ensuring that the mass of each material is conserved and that the location of the interface will coincide with jumps in density and other fluid properties, they handle changes in the topology of the front without an increase in algorithmic complexity or computational cost and the work required to update the front is small compared to the work required to update the underlying velocity field.

The remainder of this paper is organized as follows. In §2 we describe six volume-of-fluid interface reconstruction algorithms, including the two new second-order accurate algorithms. In §3 we study the spatial accuracy of these methods by using them to reconstruct various stationary interfaces. In §4 we discuss operator split advection algorithms and study the time accuracy of second-order accurate, operator split advection by using it, in combination with each of the six interface reconstruction algorithms, to approximate various interface shapes undergoing translation and rotation. In §5 we describe a second-order accurate, unsplit advection algorithm we have developed and examine the accuracy of this algorithm by applying it to the problems studied in §4. We state our conclusions in §6.

2. Volume-of-Fluid Interface Reconstruction Algorithms

In this section we consider the following problem. Let Ω be a region in the plane $I\!R^2$ and let $\mathbf{z}(s) = (x(s), y(s))$ for $0 \leq s \leq 1$ be a piecewise smooth interface. In all of the examples we study $\mathbf{z}$ is C^0 and piecewise C^2. In addition, in all but one of these examples $\mathbf{z}$ is a closed curve $\mathbf{z}(0) = \mathbf{z}(1)$, the exception being when $\mathbf{z}$ is a line, in which case $\mathbf{z}(0)$, $\mathbf{z}(1) \in \partial\,\Omega$. We think of $\mathbf{z}$ as separating Ω into two regions of fluids which we refer to as "light" and "dark" fluid. Now cover Ω with a square grid Ω^h where h denotes the grid spacing. For each $0 \leq i \leq M$ and $0 \leq j \leq N$ let $f_{i,j}$ represent the fraction of the i,jth cell's volume that is occupied by the dark fluid. The problem is to reconstruct the interface $\mathbf{z}$, given only the grid Ω^h and the volume fractions $f_{i,j}$, $i = 0, ..., M$ and $j = 0, ..., N$. We refer to an algorithm for solving this problem as a volume-of-fluid *interface reconstruction algorithm*.

Each of the algorithms described in this section produces a linear approximation to the interface in each multifluid cell; i.e., each cell which satisfies $0 < f_{i,j} < 1$. (We use the terms multi-fluid and multi-material interchangeably). In general, these piecewise linear approximations are not continuous. All of the algorithms except for SLIC use the volume fractions in a 3×3 block of cells to determine the approximate interface in the center cell of the block. SLIC uses only the volume fractions in a 3×1 block of cells to determine the approximate interface in the center cell of the block.

All of the algorithms described below except for SLIC also return a slope, or equivalently, a vector $\mathbf{n}$ normal to the interface. In this article we adopt the convention that $\mathbf{n}$ always points away from the dark fluid. The normal vector $\mathbf{n}_{i,j}$ in the i,jth cell together with the volume fraction $f_{i,j}$ uniquely determines the approximate linear interface in that cell. Thus, since the volume fraction $f_{i,j}$ is given, all of the algorithms described below (but SLIC) are simply rules for determining a unit normal vector from the values of the volume fractions in some neighborhood of the i,jth cell.

In what follows we often will replace the problem of finding the unit normal $\mathbf{n}$ to the approximate interface with that of finding its slope $\widetilde{m}$, since for many of the interface reconstruction algorithms we study this results

in very simple formulas for $\widetilde{m}$. However, this approach is problematic when the best linear approximation is a vertical, or nearly vertical, line. This can be remedied by rotating the 3×3 block of cells by 90° and applying the interface reconstruction algorithm in the new coordinate frame. In our discussion of the various interface reconstruction algorithms below we will sometimes omit details related to this coordinate transformation.

Because of the difficulty in representing a vertical line in slope intercept form, we have found that in practice it is usually preferable to represent the approximate interface in each multi-material cell as a unit vector $\mathbf{n} = (n_x, n_y)$ normal to the approximate interface together with it's distance d from the origin. In this case the line satisfies the following equation

$$n_x\, x \;+\; n_y\, y \;=\; d\,.$$

We have found that this is a better computational representation than the slope intercept form

$$y = \widetilde{m}x + b\,.$$

2.2 Simple Line Interface Calculation (SLIC) This algorithm is due to Noh and Woodward [38]. Their version of SLIC is a strictly one-dimensional method in which one uses the information in a 3×1 block of cells to reconstruct the interface in the middle cell. This necessitates the use of an operator split advection algorithm (described in §4) when one is solving problems in two and three space dimensions. Chorin [8] (see also [64]) has proposed a variant of the original SLIC algorithm that uses the volume fraction information in a 3×3 block of cells to reconstruct the interface in the center cell. However in general this modified algorithm still does not yield an approximation to the interface that is independent of the sweep direction and hence one is still constrained to use an operator split advection algorithm. Here we study the original version of SLIC as described in [38].

In the SLIC method the reconstructed interface is composed of one or (in two cases) two lines aligned with the grid. The interface geometry and location is based on the values of the volume fractions in the a row of three cells centered on the cell containing the interface. Fig. 2.1 shows the interface geometry in four of the nine possible cases. The other five cases are obtained by switching light and dark fluid, or by switching left and right. Note that the approximate interface is *not* necessarily perpendicular to the sweep direction. Since SLIC always returns horizontal or vertical lines, it obviously does not exactly reproduce all linear interfaces and hence it is at best first-order accurate.

2.3 The Center of Mass Algorithm This method is due to Saltzman [51]. In the Center of Mass algorithm, one considers the dark fluid to have a mass density of 1, and the light fluid to have no mass. To determine the approximate interface in the center of a 3×3 block of cells one first determines the center of mass $(\bar{x}, \bar{y})$ of the 3×3 block and then finds a unit vector that points from this point to the center of the center cell. This vector is taken as the unit normal $\mathbf{n}$ to the approximate interface.

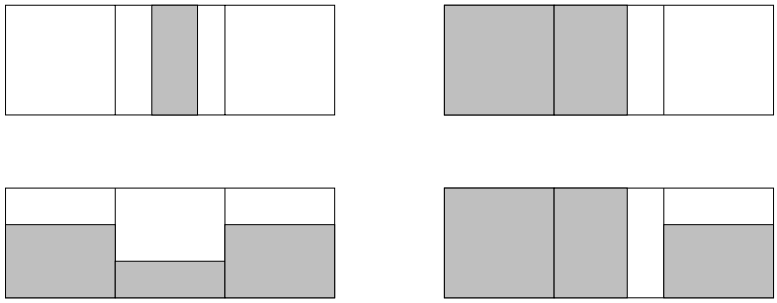


Figure 2.1 Four of the nine possible cases in SLIC. The other five cases are obtained by switching light and dark, or by switching left and right.

To see if this method reproduces all lines exactly we consider the exact version of the method. In other words, to find $(\bar{x}, \bar{y})$ we integrate exactly rather than by using a numerical approximation to the integrals as one does in practice. Let h be the cell width of the 3×3 block, and choose a coordinate system in which the center of the center cell is at the origin. If the Center of Mass algorithm reproduces all lines exactly, then in particular for arbitrary m it must reproduce the line $y = mx$ exactly. Let $(\bar{x}, \bar{y})$ be the coordinate of the center of mass of this 3×3 grid. We can find $(\bar{x}, \bar{y})$ by

$$\begin{aligned}
\bar{x} &= \frac{\int_{-1.5h}^{1.5h} \int_{-1.5h}^{mx} x \, dy \, dx}{\int_{-1.5h}^{1.5h} \int_{-1.5h}^{mx} dy \, dx} = \frac{\int_{-1.5h}^{1.5h} mx^2 + 1.5hx \, dx}{\int_{-1.5h}^{1.5h} mx + 1.5h \, dx} = \frac{2.25mh^3}{4.5h^2} = \frac{mh}{2}, \\
\bar{y} &= \frac{\int_{-1.5h}^{1.5h} \int_{-1.5h}^{mx} y \, dy \, dx}{\int_{-1.5h}^{1.5h} \int_{-1.5h}^{mx} dy \, dx} = \frac{\int_{-1.5h}^{1.5h} m^2x^2 - 2.25h^2 \, dx}{9h^2} = \frac{2.25m^2h^3 - 6.75h^3}{9h^2} = \frac{m^2h - 3h}{4}.
\end{aligned}$$

The slope of the line given by this method is found by

$$\frac{\bar{x}}{-\bar{y}} = \frac{mh}{2} \frac{4}{3h - m^2h} = \frac{2m}{3 - m^2}.$$

Thus the center of mass algorithm does not exactly reconstruct the line $y = mx$, and hence it is at best first-order accurate.

2.4 Parker and Youngs' Method In this method, due to Parker and Youngs [41], one calculates an approximation to ∇f, which is taken to point in the direction normal to the approximate interface. One calculates ∇f with the following difference scheme

$$\begin{aligned}
\frac{\partial f}{\partial x} &= \frac{f_E - f_W}{2}, \\
\frac{\partial f}{\partial y} &= \frac{f_N - f_S}{2}.
\end{aligned}$$

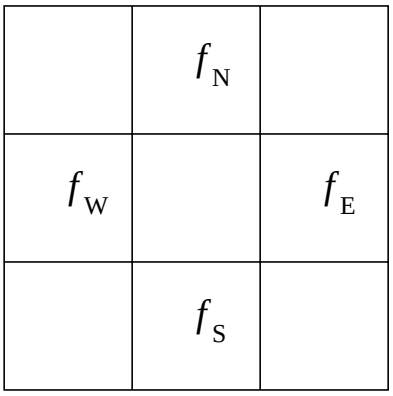


Figure 2.2 The stencil that Parker and Youngs use to determine ∇f.

The variables f_E, f_W, f_N, f_S are centered in the cells as shown in Fig. 2.2 and are given by

$$\begin{aligned} f_E &= \frac{1}{2+\alpha}(f_{i+1,j-1} + \alpha f_{i+1,j} + f_{i+1,j+1}) , \\ f_W &= \frac{1}{2+\alpha}(f_{i-1,j-1} + \alpha f_{i-1,j} + f_{i-1,j+1}) , \\ f_N &= \frac{1}{2+\alpha}(f_{i-1,j+1} + \alpha f_{i,j+1} + f_{i+1,j+1}) , \\ f_S &= \frac{1}{2+\alpha}(f_{i-1,j-1} + \alpha f_{i,j-1} + f_{i+1,j-1}) , \end{aligned}$$

where α is a free parameter. Parker and Youngs report that $\alpha = 2$ seems to give the best results.

In order to determine how well Parker and Youngs' method approximates straight lines, we consider the line $y = \frac{1}{3}x + h$ shown in Fig. 2.3a. The volume fractions due to this line are shown in Fig. 2.3b. The values of f_E, f_W, f_N, f_S are

$$\begin{aligned} f_E &= \frac{1}{\alpha+2}\,(\frac{5}{6}\alpha + 1) , \\ f_W &= \frac{1}{\alpha+2}\,(\frac{1}{6}\alpha + 1) , \\ f_N &= 0 , \\ f_S &= \frac{1}{\alpha+2}\,(1 + \alpha + 1) \;=\; 1 , \end{aligned}$$

and hence

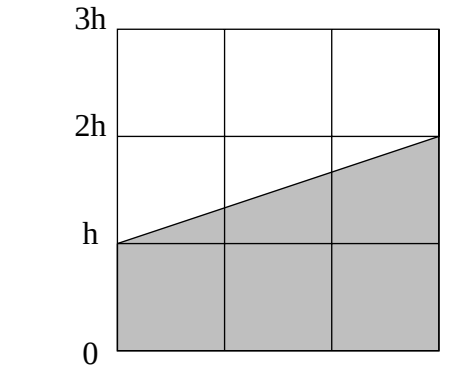
$$\begin{aligned} \frac{\partial f}{\partial x} &= \frac{(f_E - f_W)}{2} \;=\; \frac{\alpha}{3(\alpha+2)} , \\ \frac{\partial f}{\partial y} &= \frac{(f_N - f_S)}{2} \;=\; -\frac{1}{2} . \end{aligned}$$

The slope of the approximate interface is therefore

$$\widetilde{m} \;=\; \frac{-\partial f/\partial x}{\partial f/\partial y} \;=\; \frac{2\alpha}{3(\alpha+2)} .$$

The correct slope of the line is $m = 1/3$. Thus if we wish to choose α so that

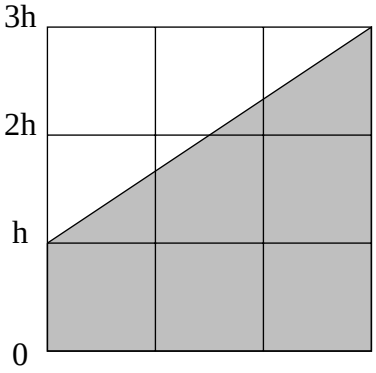
$$\frac{2\alpha}{3(\alpha+2)} \;=\; \frac{1}{3} .$$



(a)

0	0	0
1/6	1/2	5/6
1	1	1

(b)



(c)

0	1/12	2/3
1/3	11/12	1
1	1	1

(d)

Figure 2.3 (a) Parker and Youngs' method will reconstruct this line exactly only if $\alpha = 2$. (b) The volume fractions associated with the line shown in (a). (c) Parker and Youngs' method does not reconstruct this line exactly for $\alpha = 2$. Thus it does not reproduce all linear interfaces exactly, and so we conclude it is a first-order method. (d) The volume fractions associated with the line shown in (c).

we must have $\alpha = 2$. In other words, only the value of $\alpha = 2$ will yield the correct linear interface $y = \frac{1}{3}x + h$.

We now show that $\alpha = 2$ does not reconstruct all lines exactly. Consider the line $y = \frac{2}{3}x + h$ shown in Fig. 2.3c. The volume fractions due to this line are shown in Fig. 2.3d. When $\alpha = 2$ the values of f_E, f_W, f_N, f_S are

$$\begin{aligned} f_E &= \frac{1}{4}(1 + 2 + \frac{2}{3}) = \frac{11}{12}, \\ f_W &= \frac{1}{4}(1 + \frac{2}{3} + 0) = \frac{5}{12}, \\ f_N &= \frac{1}{4}(0 + \frac{2}{12} + \frac{2}{3}) = \frac{5}{24}, \\ f_S &= \frac{1}{4}(1 + 2 + 1) = 1, \end{aligned}$$

and hence

$$\begin{aligned} \frac{\partial f}{\partial x} &= \frac{1}{2}(\frac{11}{12} - \frac{5}{12}) = \frac{1}{4}, \\ \frac{\partial f}{\partial y} &= \frac{1}{2}(\frac{5}{24} - 1) = \frac{-19}{48}. \end{aligned}$$

The slope of the approximate interface is therefore

$$\widetilde{m} \;=\; \frac{-\partial f/\partial x}{\partial f/\partial y} \;=\; \frac{1}{4}\frac{48}{19} \;=\; \frac{12}{19}\,. \tag{2.1}$$

Since the correct slope is $m = 2/3$, we conclude that Parker and Youngs' algorithm will not reconstruct all linear interfaces exactly.

Note that the quantity in (2.1) is independent of the grid width h. This implies that the approximation to the slope does not improve as $h \to 0$; i.e., in general this algorithm makes an $O(1)$ error in the slope of the interface. We therefore conclude that it is at best a first-order accurate algorithm. This is consistent with the numerical results presented in §4.

2.5 The Least Squares Volume-of-Fluid Interface Reconstruction Algorithm (LVIRA) This algorithm is due to Puckett [46] and has been used extensively to model gas interfaces in compressible (e.g., see [35, 36, 48, 49]) and incompressible (e.g., see [3, 47]) flows. Consider the 3×3 block of cells centered on the i,jth cell. Let $f(x)$ be a curve that passes through the i,jth cell and let $f_{k,l}$ for $k = i-1, ..., i+1$, $l = j-1, ..., j+1$ represent the volume fractions due to the function f, in the 3×3 block. Now let $\tilde{f}$ be a linear approximation to f with slope $\widetilde{m}$ and volume fractions $\tilde{f}_{k,l}$ and assume that $\tilde{f}$ has the same volume fraction in the i,jth cell as f; i.e., $f_{i,j} = \tilde{f}_{i,j}$. Define $E^2_{i,j}$ to be the discrete L^2 error between the volume fractions in the 3×3 block of cells centered on the i,jth cell,

$$E^2_{i,j}(\widetilde{m}) \;=\; \Big(\sum_{k,l=-1}^{1} (\tilde{f}_{i+k,j+l}(\widetilde{m}) \;-\; f_{i+k,j+l})^2 \Big)^{\frac{1}{2}}\,. \tag{2.2}$$

In the LVIRA algorithm one minimizes $E^2_{i,j}$ as a function of $\widetilde{m}$ by rotating the the line $\tilde{f}$ under the constraint that this line exactly reproduces the volume fraction in the center cell, $\tilde{f}_{i,j} = f_{i,j}$.[3]

Note that basic design criterion in the LVIRA algorithm is to minimize some measure of the error between the volume fractions given by the true and approximate interfaces. One could instead choose to minimize the discrete L^∞ error

$$E^\infty_{i,j}(\widetilde{m}) \;=\; \max_{k,l=-1,1} |\tilde{f}_{i+k,j+l}(\widetilde{m}) \;-\; f_{i+k,j+l}| \tag{2.3}$$

or the discrete L^1 error

$$E^1_{i,j}(\widetilde{m}) \;=\; \sum_{k,l=-1}^{1} |\tilde{f}_{i+k,j+l}(\widetilde{m}) \;-\; f_{i+k,j+l}| \tag{2.4}$$

in the 3×3 block of cells centered on the i,jth cell, subject to the constraint that $\tilde{f}_{i,j} = f_{i,j}$.

[3] In order for (2.2) to represent our algorithm correctly one must allow $\widetilde{m}$ to have the value $\widetilde{m} = \infty$. For this reason it is perhaps better to express the error $E_{i,j}$ in (2.2) as a function of the unit normal $\mathbf{n}$ to the approximate interface. However, we find that the formulas for the approximate slope $\widetilde{m}$ are much simpler to write down and understand. We hope that the use of $\widetilde{m}$ in (2.2) this will not cause the reader confusion.

We claim that if the original interface $f(x)$ is a line, then the LVIRA algorithm - with any of the norms defined in (2.2)-(2.4) - will exactly reconstruct this line in the i,jth cell. To see this suppose that $f(x)$ is a line and assume that that the minimization procedure will always find the correct global minimum when given volume fraction data $f_{k,l}$ due to a linear interface in a 3×3 block of cells. (Our test problems below demonstrate that this is a reasonable assumption.) Each of the norms $E^{\alpha}_{i,j}$ in (2.2)-(2.4) has a minimum value of 0 that is attained when $f_{k,l} = \tilde{f}_{k,l}$ for each cell in the 3×3 block. This will only occur when $\tilde{f}(x) = f(x)$. Thus the LVIRA algorithm reconstructs linear interfaces exactly.

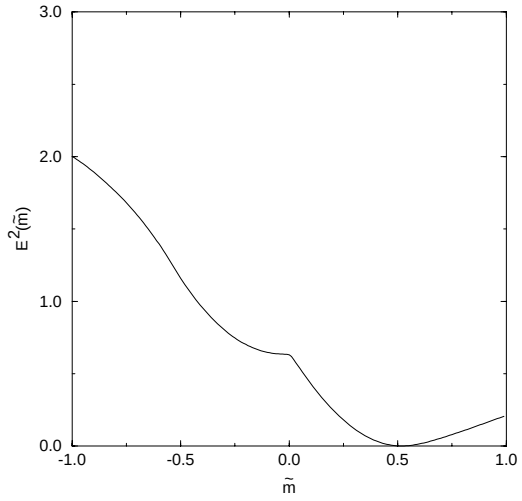


Figure 2.4 The error $E^2_{i,j}(\widetilde{m})$ between a circle that passes through the cell center, with a tangent line of slope of 0.5 at that point, and the approximate interface $\tilde{f}$ with slope $\widetilde{m}$.

In the work presented below we determine the slope $\widetilde{m}$ by using the central difference algorithm to obtain an initial guess and then using Brent's algorithm (e.g., see [44]) to minimize $E^2_{i,j}$. (Brent's algorithm is an iterative method that fits a parabola over the interval, and uses the minimum of the parabola as the next guess for the minimum of the given function. If it cannot fit a parabola, it does a golden section search. The method stops when both the interval and consecutive guesses are within a given tolerance.) To help ensure that Brent's method will converge to the global minimum, we slowly expanded the interval about the initial guess until the error at the endpoints of the interval was greater than the one given by the initial guess. In Fig. 2.5 we present an example of Brent's method improving on the initial guess given by the central difference algorithm and finding the minimum of the function $E^2_{i,j}(\widetilde{m})$ shown in Fig. 2.4.

The term "Least Squares" that has come to be associated with this algorithm may be somewhat misleading. It was chosen because the method was originally designed to minimize the discrete L^2 error defined in (2.2). Since this is the same measure of error that is minimized in a "least squares" data fit it seemed

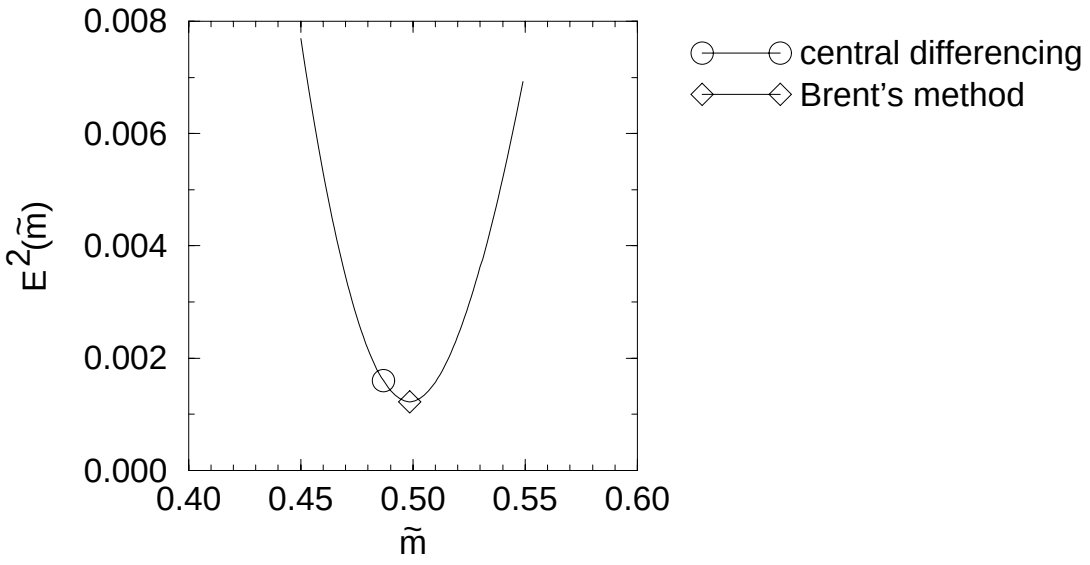


Figure 2.5 We use the centered difference algorithm to obtain a starting point for Brent's method, which we then use to find the minimum of the curve shown in Figure 2.4.

natural to refer to the algorithm as the "Least Squares" Volume-of-Fluid Interface Interface Reconstruction Algorithm. However, one should note that given a *fixed* volume fraction $f_{i,j}$ in the center cell, the function $\mathcal{F}(\mathbf{n})$ that takes a unit vector $\mathbf{n}$ normal to the approximate linear interface $\tilde{f}$ in the center cell and returns the volume fractions $\tilde{f}_{i+k,j+l}$ for $k,l = -1, ..., 1$ in the 3×3 block of cells surrounding this cell, subject to the constraint that $\tilde{f}_{i,j} = f_{i,j}$, is nonlinear. Thus, unlike the least squares data fitting algorithm, the problem of minimizing (2.2) can not be formulated as the solution of a system of linear equations.

2.6 Efficient Least Squares Volume-of-Fluid Interface Reconstruction Algorithm (ELVIRA) This algorithm is due to Pilliod [42]. In the ELVIRA method, one obtains the slope $\widetilde{m}$ of the approximate linear interface $\tilde{f}$ by choosing between six candidate values of $\widetilde{m}$. The first three of these six candidate values are the backward, central and forward differences of the column sums of the volume fractions. In other words, we consider the following three values

$$\begin{aligned} \widetilde{m}_b^x &= \sum_{l=-1}^{1} f_{i,j+l} - f_{i-1,j+l} \,, \\ \widetilde{m}_c^x &= \sum_{l=-1}^{1} f_{i+1,j+l} - f_{i-1,j+l} \,, \\ \widetilde{m}_f^x &= \sum_{l=-1}^{1} f_{i+1,j+l} - f_{i,j+l} \,. \end{aligned} \tag{2.5}$$

The other three candidate values are the backward, central, and forward differences of the column sums of

the volume fractions *in the y direction*; i.e., the differences of the row sums,

$$\begin{aligned} \widetilde{m}_b^y &= \sum_{k=-1}^{1} f_{i+k,j} - f_{i+k,j-1} \,, \\ \widetilde{m}_c^y &= \sum_{k=-1}^{1} f_{i+k,j+1} - f_{i+k,j-1} \,, \\ \widetilde{m}_f^y &= \sum_{k=-1}^{1} f_{i+k,j+1} - f_{i+k,j} \,. \end{aligned} \tag{2.6}$$

(Note that the slopes $\widetilde{m}^y$ are with respect to the coordinate system which is rotated 90° from the original coordinate system. The lines $\tilde{f}(\widetilde{m}^y)$ associated with these slopes and the resulting errors $E_{i,j}^p(\widetilde{m}^y)$ must be interpreted appropriately.) To determine which is the best slope $\widetilde{m}$ to use for a given collection of volume fractions $f_{k,l}$ in the 3×3 block we minimize one of the norms in (2.2)-(2.4) over the slopes in (2.5) and (2.6),

$$\widetilde{m} = \{\widetilde{m}_\beta^\alpha : E_{i,j}^p(\widetilde{m}_\beta^\alpha) = \min[E_{i,j}^p(\widetilde{m}_b^x), ..., E_{i,j}^p(\widetilde{m}_f^y)] \text{ for } \alpha = x, y \text{ and } \beta = b, c, f\} \,. \tag{2.7}$$

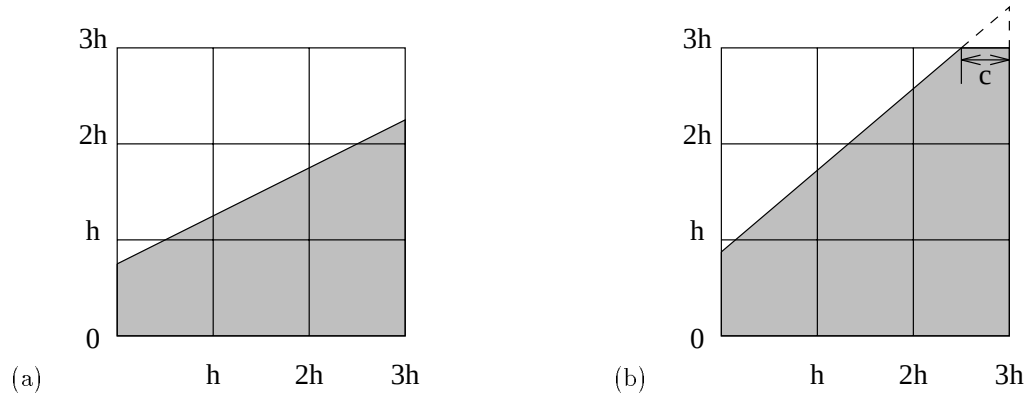


Figure 2.6 (a) Center differences will exactly reconstruct a line that cuts opposite sides of a 3×3 block of cells. (b) It will not exactly reconstruct a line that cuts adjacent sides of a 3×3 block of cells.

In order to examine how well this method approximates an arbitrary line, we must consider the following two cases: (a) the line cuts opposite sides of the 3×3 grid, as in Fig. 2.6a or (b) it cuts adjacent sides, as in Figure 2.6b. First consider the case shown in Fig. 2.6a. Suppose that the interface be given by $y(x) = mx + b$. Let A_1 be the sum of the volume fractions in the left hand column and A_3 the sum of the volume fractions in the right hand column in Fig. 2.6a. We can determine A_1 by noting that it is the area of the trapezoid with sides of length b and $mh + b$ and width h while A_3 is the area of the trapezoid with sides of length $2mh + b$ and $3mh + b$ and width h,

$$\begin{aligned} A_1 &= \frac{1}{2h^2}(mh + b + b)h = \frac{m}{2} + \frac{b}{h} \,, \\ A_3 &= \frac{1}{2h^2}(3mh + b + 2mh + b)h = \frac{5}{2}m + \frac{b}{h} \,. \end{aligned}$$

Note that these formulas are exact no matter how the line $y(x)$ intersects a given cell, provided only that the line cuts opposite sides of the 3×3 grid. The central difference approximation to $m = y'(x)$ is given by

$$\widetilde{m} \;=\; \frac{1}{2} \sum_{k\,=\,-1}^{1} f_{i+1,j+k} - f_{i-1,j+k} \;=\; \frac{A_3 - A_1}{2} \;=\; m\,.$$

Since the exact and approximate interfaces have the same slopes and the same volume fraction in the center cell, they are the same line. Thus, when the true interface is a line that intersects opposite sides of the 3×3 block, the approximate interface in the center cell will be precisely this line.

Now suppose that the linear interface does not intersects opposite sides of the 3×3 block. Therefore, it must intersect adjacent sides of the 3×3 block. Consider the case shown in Fig. 2.6b, where the interface is given by $y = mx + b$ with $m \leq 1$. Let A_1 be the sum of the volume fractions in the left hand column and A_2 be the sum of the volume fractions in the middle column. The quantities A_1 and A_2 are given by

$$\begin{aligned} A_1 \;&=\; \frac{m}{2} + \frac{b}{h}\,, \\ A_2 \;&=\; \frac{1}{2h^2}(2mh + b + mh + b)h \;=\; \frac{3}{2}m + \frac{b}{h}\,, \end{aligned}$$

and their difference is,

$$\widetilde{m}_b^x \;=\; A_2 - A_1 \;=\; m\,.$$

Thus a backward difference of the column sums exactly reconstructs a linear interface that intersects the left hand side of the 3×3 block and has slope $m \leq 1$. By considering the mirror image of Fig. 2.6b, one can see that a forward difference of the column sums will exactly reconstruct a linear interface that intersects the right hand side of the 3×3 block and has slope $m \geq -1$. If the magnitude of the slope m of the true linear interface is greater than one, then the argument above, applied to the 3×3 block in a coordinate frame that has been rotated $90°$, shows that either a backward or forward difference of the row sums will produce the correct slope. Thus, at least one of the errors $E_{i,j}^p(\widetilde{m}_\beta^\alpha)$ inside the square brackets in (2.7) will be 0, thereby guaranteeing that the ELVIRA algorithm will reconstruct all linear interfaces that pass through the center cell of the 3×3 block exactly.

If one only considers linear interfaces, the centered difference may seem redundant, since all three difference methods produce the correct slope when the linear interface intersects opposite sides of the 3×3 block. For nonlinear interfaces, however, it appears that a centered difference sometimes produces a more accurate approximation to the interface than the other two methods. For example, consider a circle that is placed in the 3×3 block such that the top of the circle intersects the center of the center cell. Then a backward difference results in a positive slope, a forward difference results in a negative slope, while a centered difference results in a zero slope. The latter is the best approximation to the slope of the tangent to the circle at the center of the center cell.

3. Volume-of-Fluid Advection Algorithms

In order to approximate solutions of the advection equation (1.7) we need an algorithm for evolving the volume fractions in time. Let $u^n_{i-\frac{1}{2},j}$ (*resp.* $v^n_{i,j-\frac{1}{2}}$) denote the value of u (*resp.* v) at the center of the left (*resp.* bottom) edge of the i,jth cell and suppose that these velocities satisfy a discrete form of (1.6),

$$\frac{(u^n_{i+\frac{1}{2},j} - u^n_{i-\frac{1}{2},j})}{\Delta x} + \frac{(v^n_{i,j+\frac{1}{2}} - v^n_{i,j-\frac{1}{2}})}{\Delta y} \approx 0\,. \tag{3.1}$$

Given an approximation to the interface in each cell for which $0 < f^n_{i,j} < 1$ we wish to determine the volume fractions $f^{n+1}_{i,j}$ at the new time $t^{n+1} = (n+1)\,\Delta t$. We refer to algorithms for doing this as *volume-of-fluid advection algorithms.*

In this article we study two types of advection algorithms. Both are based on the standard conservative finite difference update of (1.7),

$$f^{n+1}_{i,j} \;=\; f^n_{i,j} \;+\; \frac{\Delta t}{\Delta x}\,[F^n_{i-\frac{1}{2},j} - F^n_{i+\frac{1}{2},j}] \;+\; \frac{\Delta t}{\Delta y}\,[G^n_{i,j-\frac{1}{2}} - G^n_{i,j+\frac{1}{2}}]\,, \tag{3.2}$$

where $F^n_{i-\frac{1}{2},j} = (fu)^n_{i-\frac{1}{2},j}$ denotes the flux of f across the left-hand edge of the i,jth cell and $G^n_{i,j-\frac{1}{2}} = (fv)^n_{i,j-\frac{1}{2}}$ denotes the flux across the bottom edge of the i,jth cell, etc.

3.1 Operator Split Advection The simplest advection algorithm for approximating solutions of (1.7) is the *fractional step* or *operator split* method,

$$f^*_{i,j} \;=\; f^n_{i,j} \;+\; \frac{\Delta t}{\Delta x}\,[F^n_{i-\frac{1}{2},j} - F^n_{i+\frac{1}{2},j}]\,, \tag{3.3}$$

$$f^{n+1}_{i,j} \;=\; f^*_{i,j} \;+\; \frac{\Delta t}{\Delta y}\,[G^*_{i,j-\frac{1}{2}} - G^*_{i,j+\frac{1}{2}}]\,. \tag{3.4}$$

where the superscript * represents an intermediate value for the volume fractions and fluxes. There is a simple geometric interpretation of the fluxes in (4.3)-(4.4). Suppose that $u^n_{i+\frac{1}{2},j}$ is positive. Divide the (i,j)th cell into two disjoint rectangles, with areas $u^n_{i+\frac{1}{2},j}\;\Delta t\;\Delta y$ on the right and $(\,\Delta x - u^n_{i+\frac{1}{2},j}\;\Delta t\,)\;\Delta y$ on the left as shown in Fig. 3.1a.

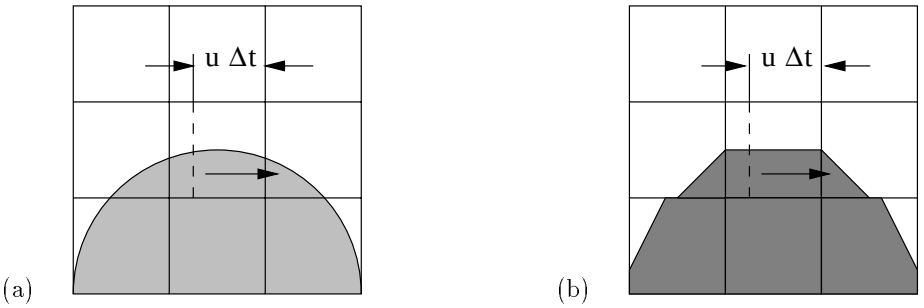


Figure 3.1 (a) In operator split advection, the fluid to the right of the dotted line crosses the right cell edge. (b) In a volume-of-fluid method, we use the reconstructed interface to determine the amount of fluid that crosses each edge.

All of the fluid to the right of the dotted line in Fig. 3.1a will cross the right-hand edge during this time step. In particular, the flux of dark fluid across this edge is equal to the amount of dark fluid contained in this rectangle. In a volume-of-fluid method, this can be determined by the location of the reconstructed interface as shown in Fig. 3.1b. Thus, if $V_{i+\frac{1}{2},j}$ denotes the volume of dark fluid in the center cell to the right of the dotted line in Fig. 3.1b, then the (approximate) volume fraction flux across the right hand cell edge is given by

$$F^n_{i+\frac{1}{2},j} \;=\; u^n_{i+\frac{1}{2},j}\, V_{i+\frac{1}{2},j} \,/ (u^n_{i+\frac{1}{2},j}\;\Delta t\;\Delta y) \;=\; V_{i+\frac{1}{2},j} \,/ (\,\Delta t\;\Delta y\,)\,. \tag{3.5}$$

After using (4.5) in (4.3) to determine the intermediate volume fractions $f^*_{i,j}$, one then uses these values to reconstruct the interface in all cells that satisfy $0 < f^*_{i,j} < 1$. The vertical fluxes $G^*_{i,j+1/2}$ are then determined by a geometric construction analogous to the one described for the horizontal fluxes, and the volume fractions at the new time level $f^{n+1}_{i,j}$ are found by inserting these vertical fluxes into (4.4). This procedure can be made second-order accurate simply by alternating the sweep direction at each time step.

The CFL Constraint It is apparent from geometric considerations that one must choose the CFL number σ so that the amount of fluid which leaves a cell in one time step is no more than the amount of fluid that was originally in the cell. In other words, one must choose σ so that

$$V_{i+\frac{1}{2},j} \;-\; V_{i-\frac{1}{2},j} \le f_{i,j}\;\Delta x\;\;\Delta y \tag{3.6}$$

for all i, j. One way to ensure that (4.6) is always satisfied is to choose $\sigma \in (0,1]$ so that

$$|u^n_{i+\frac{1}{2},j}|\;\Delta t \;\le\; \Delta x\,/2 \qquad \text{and} \qquad |v^n_{i,j+\frac{1}{2}}|\;\Delta t \;\le\; \Delta y\,/2 \qquad \text{for all}\, i\,,\, j\,. \tag{3.7}$$

An alternative, is to choose $\sigma \in (0,1]$ so that

$$\left(u^n_{i+\frac{1}{2},j} - u^n_{i-\frac{1}{2},j}\right)\,\Delta t \;\le\; \Delta x \qquad \text{and} \qquad \left(v^n_{i,j+\frac{1}{2}} - v^n_{i,j-\frac{1}{2}}\right)\,\Delta t \;\le\; \Delta y\;. \tag{3.8}$$

This latter condition is less restrictive than (4.7) and will usually result in a larger time step

3.2 A First-Order Unsplit Advection Algorithm For many problems one will obtain satisfactory results with the second-order accurate, fractional step method described in §4. However for some problems, such as unstable displacements in porous media, fractional step methods can distort the interface (e.g., see the discussion in [5]). A characteristic feature of this problem is the so-called "push-pull" or "staircase" phenomenon. For problems such as these it is preferable to use an unsplit advection algorithm. In this section we present an unsplit, volume-of-fluid advection algorithm that is based on the approach used by Bell, Dawson, and Shubin [5] to develop a second-order accurate, unsplit, finite difference method for approximating solutions of scalar hyperbolic conservation laws. We then present the results of applying this

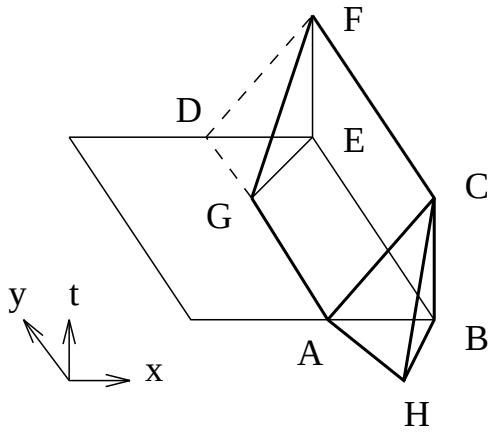


Figure 3.2 In this space-time diagram, the fluid inside the solid fluxes through the right cell edge.

advection algorithm to the test problems studied in §4. Since SLIC is an inherently one-dimensional method, we will not use it in this section.

In order to present the basic idea behind the unsplit algorithm we begin by describing a first-order accurate version. We wish to use a conservative finite difference method of the form (4.2) to approximate solutions of the conservation law (1.7). To illustrate our approach we assume that $u > 0$ and $v > 0$, and describe how one determines the flux $F^n_{i+\frac{1}{2},j}$. The other cases are analogous. The flux through the right-hand edge of the (i,j)th cell in the time interval (t^n, t^{n+1}) is

$$\begin{aligned} F^n_{i+\frac{1}{2},j} &= \int_{t^n}^{t^{n+1}} \int_{y_{i,j-\frac{1}{2}}}^{y_{i,j+\frac{1}{2}}} u(x,y,t)\, f(x_{i+\frac{1}{2},j}, y, t)\, dy\, dt \\ &= u_{i+\frac{1}{2},j} \int_{t^n}^{t^{n+1}} \int_{y_{i,j-\frac{1}{2}}}^{y_{i,j+\frac{1}{2}}} f(x_{i+\frac{1}{2},j}, y, t)\, dy\, dt\,. \end{aligned} \tag{3.9}$$

where we have assumed that in our numerical discretization $u_{i+\frac{1}{2},j}$ is constant on the space time interval $(y_{i,j-\frac{1}{2}}, y_{i,j+\frac{1}{2}}) \times (t^n, t^{n+1})$. This integral is the amount of dark fluid in the space-time rectangle $BCEF$ shown in Fig. 3.2.

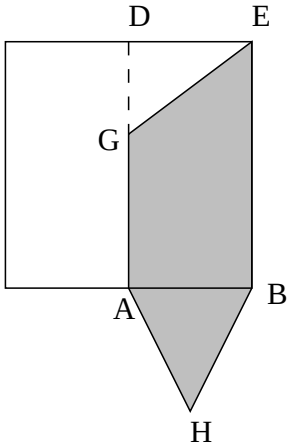


Figure 3.3 Domain of dependence for characteristics passing through the right edge of the cell (i,j).

We can find this amount by tracing back along the characteristics that originate from the rectangle $BCEF$. This gives us the solid region $ABCEFGH$ shown in Fig. 3.2. The domain of dependence of these characteristics at time t^n is the shaded region in Fig. 3.3.

Note that this region is the rectangle $ABDE$, plus the triangle ABH, minus the triangle DEG,

$$F^n_{i+\frac{1}{2},j} \approx \int\int_{ABDE} f\,dx\,dy + \int\int_{ABH} f\,dx\,dy - \int\int_{DEG} f\,dx\,dy \tag{3.10}$$

In order to approximate the right hand side of (5.10) we use one of the volume-of-fluid interface reconstruction algorithms described in §2 to determine an approximation to the interface in cell (i,j) and cell $(i,j-1)$. We then compute the area of the intersection of the dark fluid with the rectangle $ABDE$ and the triangles DEG and ABH. This yields an approximation to each of the terms on the right hand side of (5.10) and hence an approximation to $F^n_{i+\frac{1}{2},j}$. The fluxes through the other three edges of the cell are found in an analogous manner. This method for calculating the flux, which Colella [9] calls Corner Transport Upwind (CTU) is first-order accurate. See [9] for a discussion of the accuracy of this method and [5] and [30] for the results of tests when this method is implemented as a finite difference algorithm.

3.3 A Second-Order Unsplit Advection Algorithm We now describe a second-order, unsplit, volume-of-fluid advection algorithm. We approximate the flux $F^n_{i+\frac{1}{2},j}$ in (3.9) by integrating (1.7) over the prism $ABCDEF$ and integrating by parts. Writing (1.7) in the form

$$f_t + u_x\,f + u\,f_x + (v\,f)_y = 0\,, \tag{3.11}$$

and setting $u = u_{i+\frac{1}{2},j}$ and $u_x = (u_x)_{i,j}$ we find that,

$$\int\int\int_{ABCDEF} (f_t + (u_x)_{i,j}\,f + u_{i+\frac{1}{2},j} f_x + (vf)_y)\,dx\,dy\,dt = 0\,. \tag{3.12}$$

Integrating the above expression by parts, and noting that $u_{i+\frac{1}{2},j}$ is constant, we find that the flux $F^n_{i+\frac{1}{2},j}$ is given by

$$\begin{aligned} F^n_{i+\frac{1}{2},j} = \int\int_{ABDE} f\,dx\,dy + \int\int_{ABC} v\,f\,dx\,dt \\ - \int\int_{DEF} v\,f\,dx\,dt + \int\int\int_{ABCDEF} (u_x)_{i,j}\,f\,dx\,dy\,dt\,. \end{aligned} \tag{3.13}$$

The integral over $ABDE$ is the volume of dark fluid in this rectangle. As above we use an interface reconstruction algorithm to determine an approximation to the interface in the (i,j)th cell and use this approximation to compute the area of the intersection of the dark fluid with rectangle $ABDE$ to compute this quantity.

Now let R_1 be the ratio of the volume of dark fluid in $ABDE$ to the area of $ABDE$, and let V_1 be the volume of the prism $ABCDEF$. We approximate the volume integral in (3.13) by

$$\int\int\int_{ABCDEF} (u_x)_{i,j}\,f\,dx\,dy\,dt \approx R_1\,V_1\,(u_x)_{i,j}\,.$$

In order to evaluate the integral over DEF we integrate (1.7) in the form

$$\int\int\int_{ABCDEF} (f_t + (u_x)_{i,j}\, f \,+\, u_{i+\frac{1}{2},j} f_x + (vf)_y)\, dx\, dy\, dt \,=\, 0\,. \tag{3.14}$$

The domain of dependence of DEF is the triangle DEG. The tetrahedron $DEFG$ is related to the triangle DEF through

$$\int\int_{DEF} f\, dx\, dt \,=\, \int\int_{DEG} f\, dx\, dy \,+\, \int\int\int_{DEFG} ((u_x)_{i,j} \,+\, (v_y)_{i,j})\, f\, dx\, dy\, dt\,. \tag{3.15}$$

The integral over DEG is the volume of dark fluid in the triangle DEG We approximate this quantity by using an interface reconstruction algorithm to determine an approximation to the interface in the (i,j)th cell and then computing the area of the intersection of the dark fluid with triangle DEG.

Let R_2 be the ratio of the volume of dark fluid in DEG to the area of DEG, and let V_2 be the volume of the tetrahedron $DEFG$. Then the volume integral in (3.15) is approximately

$$\int\int\int_{DEFG} \big((u_x)_{i,j} \,+\, (v_y)_{i,j}\big)\, f\, dx\, dy\, dt \,=\, R_2\, V_2 \big((u_x)_{i,j} \,+\, (v_y)_{i,j}\big)\,.$$

We evaluate integral over ABC in a similar manner. Thus we are able to evaluate each term of (3.13), and hence determine the flux of dark fluid through the right edge of the cell.

Note that if $v_{i,j+\frac{1}{2}} < 0$, then the point G will lie in the $(i,j+1)$th cell. Thus the tetrahedron $DEFG$ will lie in the $(i,j+1)$th cell instead of the (i,j)th cell. In this case we add the tetrahedron $DEFG$ to the prism $ABCDEF$, instead of subtracting it as we did above. Hence we add the integral over DEF instead of subtracting it. In order to avoid the distorted region that arises when $u_{i+\frac{1}{2},j}$ and $u_{i+\frac{1}{2},j+1}$ are of opposite sign we determine the x-coordinate of the vertex G by

$$x \,=\, \min(i\,\Delta x \,+\, \frac{\Delta x}{2}, i\,\Delta x \,+\, \frac{\Delta x}{2} - \Delta t\, u_{i+\frac{1}{2},j+1})\,.$$

In this way we are assured that G lies in the $(i,j+1)$th cell.

4. Test Problems

4.1 Split Operator Test Problems We begin by studying the accuracy with which second-order operator splitting combined with each of the interface reconstruction methods approximates a line that is translating in a constant velocity field. We obtained the errors reported in Table 4.1 by translating 100 randomly generated lines with unit velocity in a randomly generated direction for one unit of time and averaging the error E^1 between the approximate and exact solutions. It is apparent from the data in Table 4.1 that the SLIC, Center of Mass, centered difference, and Parker & Youngs' algorithms are all first-order accurate. As with a stationary line, the LVIRA and ELVIRA methods essentially reproduce the interface exactly. The

	1/16	1/32	rate	1/64	rate
SLIC	1.2E-1	6.6E-2	0.91	3.0E-2	1.10
Center of Mass	1.8E-2	9.5E-3	0.95	4.4E-3	1.08
Parker & Youngs	3.3E-3	1.9E-3	0.87	8.4E-4	1.13
LVIRA	4.0E-13	2.1E-13	N/A	8.2E-14	N/A
ELVIRA	9.0E-17	4.2E-17	N/A	7.9E-17	N/A

Table 4.1 The average E^1 error after translating 100 randomly generated lines one unit in time.

error given by the LVIRA method is due to the tolerance we used in Brent's algorithm. The ELVIRA method is accurate to machine zero.

We note that one can use this as a design criterion for constructing a formally second-order accurate interface tracking algorithm. Namely that it must propagate a straight line with any slope, in a uniform velocity field in any direction, exactly.

Next we present three tests with circles. In the first test we translate a unit circle in the x-direction with unit velocity for one unit of time using various CFL numbers σ. In Table 4.2 we present the errors when we use $\sigma = 1$ while in Table 4.3 we present the errors when we use $\sigma = 1/32$. It is apparent from the data presented in these two tables that, in general, decreasing the CFL number did not reduce the error. In fact, the amplitude of the error is generally larger when $\sigma = 1/32$ than when $\sigma = 1$, although in both cases the error decreases at the same rate. The increase in the amplitude of the error seen in Table 4.3 is almost certainly due to the accumulation of local truncation error over 32 times as many time steps. Unless noted otherwise, we set $\sigma = 0.5$ in all of the remaining test problems.

	1/16	1/32	rate	1/64	rate
SLIC	9.1E-3	4.7E-3	0.97	2.4E-3	0.98
Center of Mass	2.6E-3	1.4E-3	0.93	7.5E-4	0.93
Parker & Youngs	8.7E-4	4.6E-4	0.95	2.3E-4	1.00
LVIRA	2.8E-4	6.6E-5	2.12	1.6E-5	2.06
ELVIRA	1.6E-4	4.0E-5	2.00	1.0E-5	2.00

Table 4.2 The E^1 error after translating a unit circle one unit in time with CFL number $\sigma = 1$.

	1/16	1/32	rate	1/64	rate
SLIC	9.1E-3	4.7E-3	0.97	2.4E-3	0.98
Center of Mass	3.0E-3	1.6E-3	0.94	8.0E-4	1.00
Parker & Youngs	1.3E-3	6.6E-4	0.98	3.2E-4	1.03
LVIRA	5.0E-4	1.5E-4	1.67	4.8E-5	1.56
ELVIRA	4.4E-4	1.3E-4	1.69	3.8E-5	1.71

Table 4.3 The E^1 error after translating a unit circle one unit in time with $\sigma = 1/32$.

Next we translate 100 unit circles with randomly generated centers in a randomly generated direction with unit velocity for one unit of time. In Table 4.4 we present the E^1 error averaged over 100 randomly chosen circles. It is apparent that the Center of Mass, SLIC, and Parker & Youngs' methods are first-order accurate, and the LVIRA and ELVIRA methods are second-order accurate. Central difference is second-order accurate until the grid spacing is $\Delta x = 1/16$, and then it is first-order accurate.

	1/16	1/32	rate	1/64	rate
SLIC	1.8E-2	1.1E-2	0.82	6.3E-3	0.87
Center of Mass	4.8E-3	2.4E-3	1.00	1.2E-3	1.00
Parker & Youngs	1.1E-3	5.8E-4	0.95	2.8E-4	1.04
LVIRA	4.1E-4	1.1E-4	1.86	2.7E-5	2.04
ELVIRA	2.5E-4	6.6E-5	1.89	2.0E-5	1.65

Table 4.4 The average E^1 error after translating 100 random unit circles in random directions.

In our final test with circles we place a unit circle with its center at cell center and rotate it with unit angular velocity for one rotation. Here we used a CFL number of $\sigma = \pi/6$. It is apparent from the data presented in Table 4.5 that SLIC, the Center of Mass and Parker & Youngs' algorithms exhibit first-order accuracy while the other three algorithms exhibit second-order accuracy. Starting with the left-hand column and moving right, the overall decrease in the error for each algorithm when the grid was reduced from $h = 1/2$ to $h = 1/64$ was 20, 47, 185, 827, 893 and 1146 respectively. A precisely second-order accurate decrease in the error would be by a factor of 1024.

	1/16	1/32	rate	1/64	rate
SLIC	1.9E-2	8.8E-3	1.08	4.6E-3	0.96
Center of Mass	1.1E-3	6.2E-4	0.89	3.1E-4	1.00
Parker & Youngs	4.0E-4	2.0E-4	1.00	7.8E-5	1.28
LVIRA	2.5E-4	5.9E-5	2.12	1.5E-5	1.97
ELVIRA	2.3E-4	5.7E-5	2.02	1.3E-5	2.19

Table 4.5 The E^1 error after rotating a circle once with the operator split advection algorithm.

Finally we tested the second-order accurate operator split advection method with the various interface reconstruction methods on Zalezak's test problem. Here we revolved the shape shown in Fig. 4.1 about a point 5/3 units below its center for one revolution. It is apparent from the data in Table 4.6 that all of the methods exhibit an $O(h)$ decrease in the error.

In Figure 4.2 we present the results of using the operator split advection method and the various interface reconstruction methods on Zalesak's test problem on a grid with $h = 1/15$. This grid size was chosen to facillitate direct comparison with other published results of the same test problem (e.g., [5, 66]). Note that

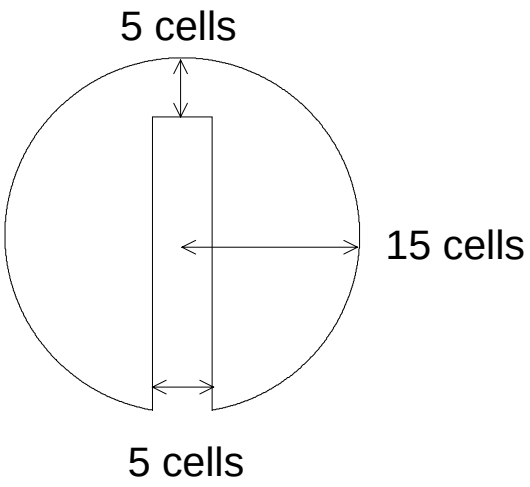


Figure 4.1 The notched circle, first introduced by Zalesak to study the accuracy of advection algorithms, which we use in several of our test problems.

the error is greatest at the corners. This is to be expected, since the interface has discontinuous derivatives at the corners. In Table 4.7 we show the difference between the initial and final area of this shape. All of the methods conserve the volume (or equivalently the mass) of the shape to machine zero.

	1/16	1/32	1/64
SLIC	-3.1E-15	-3.1E-15	7.1E-15
Center of Mass	-2.7E-15	-4.9E-15	4.4E-15
Parker & Youngs	-1.3E-15	-4.0E-15	3.1E-15
LVIRA	-1.3E-15	-2.7E-15	1.1E-14
ELVIRA	-4.4E-16	-4.0E-15	4.0E-15

Table 4.7 The difference between final and initial total area.

	1/16	1/32	rate	1/64	rate
Center of Mass	6.8E-3	3.6E-3	0.94	1.8E-3	1.00
Parker & Youngs	1.1E-2	5.6E-3	0.98	3.1E-3	0.90
LVIRA	1.1E-3	2.8E-4	1.96	7.5E-5	1.87
ELVIRA	7.7E-4	1.8E-4	2.14	5.0E-5	1.80

Table 4.8 The average E^1 error after translating 100 random unit circles in random directions.

3.2 Unsplit Operator Test Problems In this section we use the unsplit advection algorithm to compute most of the test problems presented in §4. We begin with the translation of a smooth interface, the unit circle. We take 100 unit circles with randomly generated centers, translate each circle with unit velocity in a randomly generated direction and average the error E^1 in approximating each circle.

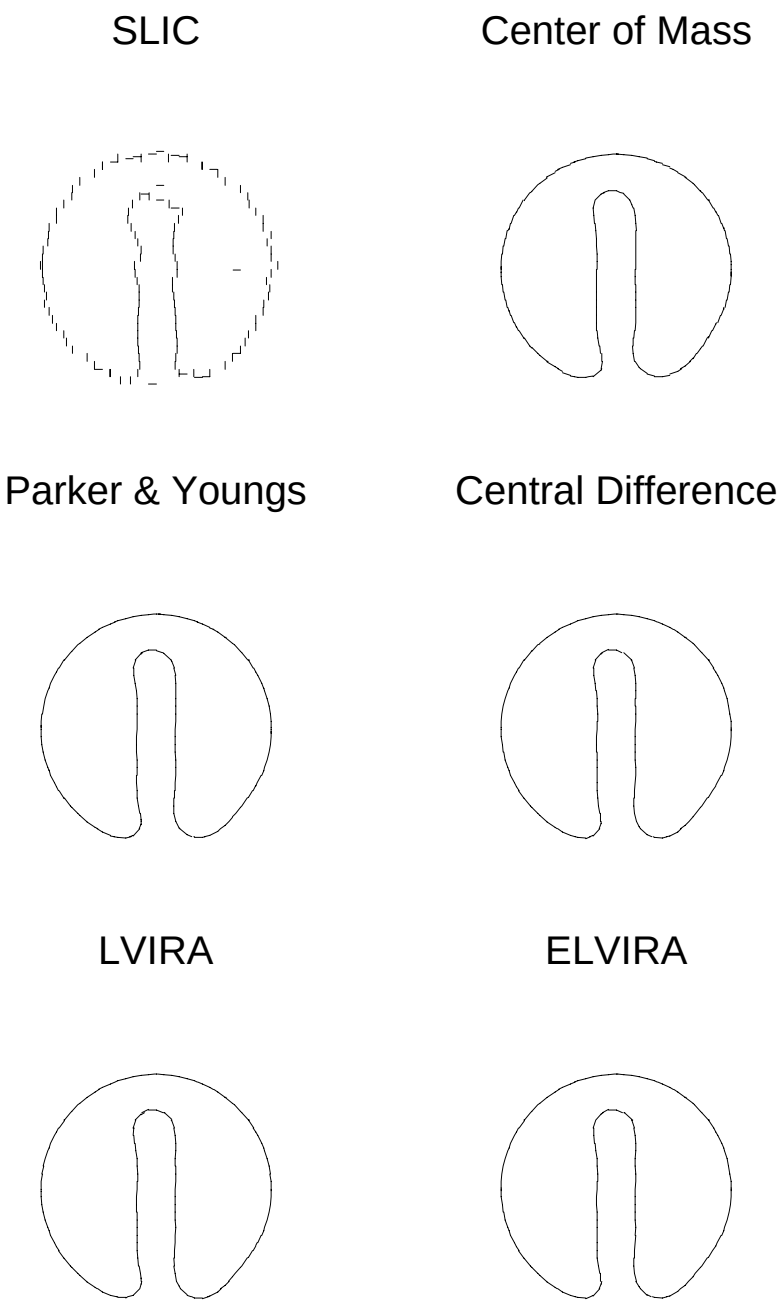


Figure 4.2 The result of using the operator split advection algorithm and the various interface reconstruction methods on Zalesak's test problem. Again notice that only the SLIC algorithm produces flotsam.

It is apparent from the data shown in Table 4.8 that the errors associated with the Center of Mass and Parker & Youngs' algorithms decrease at a rate that is not quite $O(h)$; the overall decrease is by a factor of 8 and 12 respectively. On the other hand the errors associated with the Centered Difference, LVIRA

	1/16	1/32	rate	1/64	rate
SLIC	2.5E-2	1.4E-2	0.89	6.9E-3	1.01
Center of Mass	7.2E-3	4.1E-3	0.88	2.3E-3	0.89
Parker & Youngs	6.2E-3	2.9E-3	1.07	1.4E-3	1.04
LVIRA	6.4E-3	2.9E-3	1.10	1.3E-3	1.12
ELVIRA	6.2E-3	2.8E-3	1.11	1.3E-3	1.08

Table 4.6 The average E^1 error after translating 100 random notched circles in random directions.

and ELVIRA algorithms decrease at a rate that is somewhat better than $O(h^2)$; the overall decrease in the error being 274, 276 and 376 respectively. In other words, the first two algorithms appear to be first-order accurate, while the other three appear to be second-order accurate. As in §4.3 we conjecture that the apparent second-order accurate behavior of the Centered Difference algorithm is again due to the fact that *on average* it will produce a second-order approximation to a tangent to the circle in each cell. It is important to note that this will *not* be the case is the interface is more nearly linear, in which case the Centered Difference algorithm is on average first-order accurate as shown in Tables 2.1 and 2.3.

In the next test problem we rotate a unit circle, centered on a cell center, with unit angular velocity for ten rotations. It is apparent from the data in Table 4.9 that the rate of decrease of the errors associated with each algorithm is comparable with the rate of decrease seen in the previous test problem. In this problem however, the Center of Mass and Parker and Youngs' algorithms exhibit a somewhat better than $O(h)$ decrease in the error. However, the same conclusions continue to apply.

	1/16	1/32	rate	1/64	rate
Center of Mass	1.4E-3	6.3E-4	1.11	3.0E-4	1.05
Parker & Youngs	2.9E-4	1.4E-4	1.04	6.8E-5	1.03
LVIRA	2.4E-4	5.9E-5	2.03	1.5E-5	1.97
ELVIRA	7.2E-4	1.5E-4	2.40	1.6E-5	4.69

Table 4.9 The E^1 error after rotating a circle 10 times with the unsplit advection algorithm.

In Fig. 4.4, we present five unit circles that have been rotated ten times with the various interface reconstruction methods on a grid with $h = 1/32$ and compare the results with the true solution. At this level of graphical resolution all of the approximate solutions are indistinguishable from the true solution. In fact, even when magnified by a factor of 64, the approximate interfaces still appear to be continuous - although not smooth - in spite of the fact that they are actually composed of a collection of discontinuous line segments.

Finally we test the unsplit advection method on Zalezak's test problem. As one can see from the data in Table 4.10 all of the interface reconstruction methods produce comparable errors - with the errors due

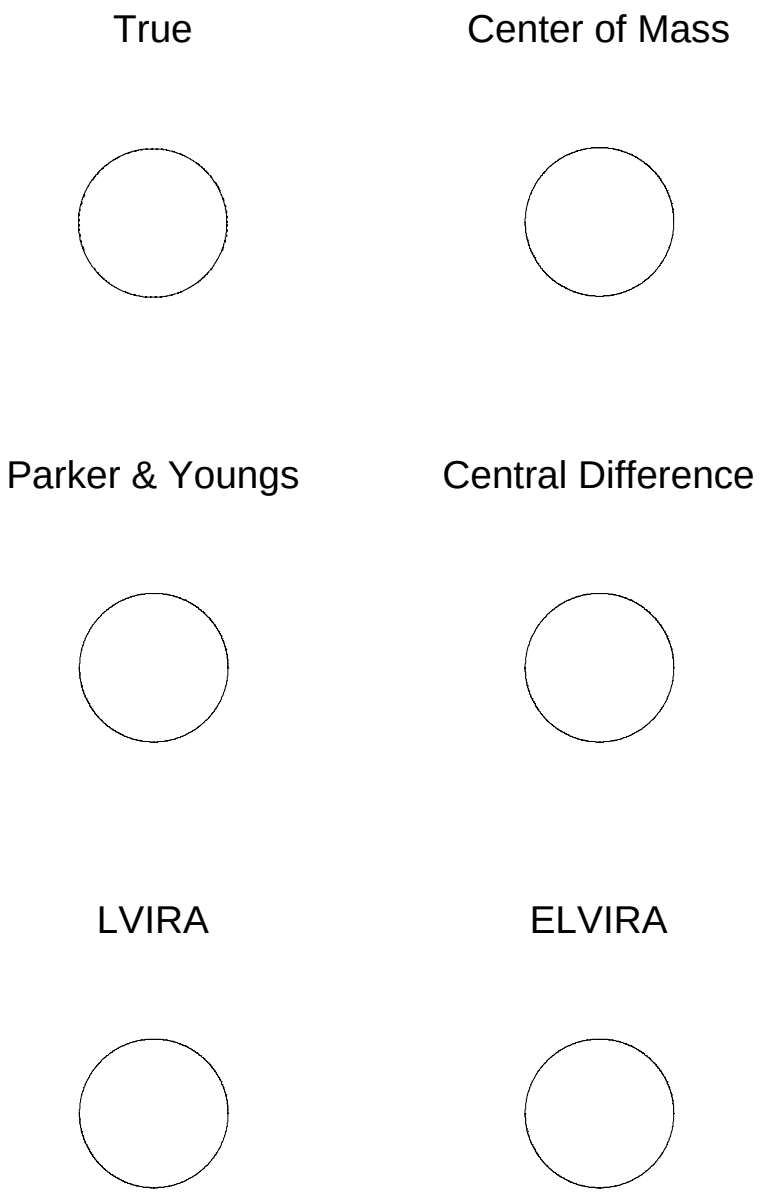


Figure 4.3 A unit circle that has been rotated for ten revolutions with the unsplit advection algorithm and various reconstruction methods.

to the Center of Mass algorithm being slightly larger than the others - and that these errors decrease at a rate that is marginally better than $O(h)$. In Table 4.11 we present the difference between the volume of the initial shape and the final shape. It is apparent that all of the methods conserve the volume (and hence the mass) of the shape to machine zero.

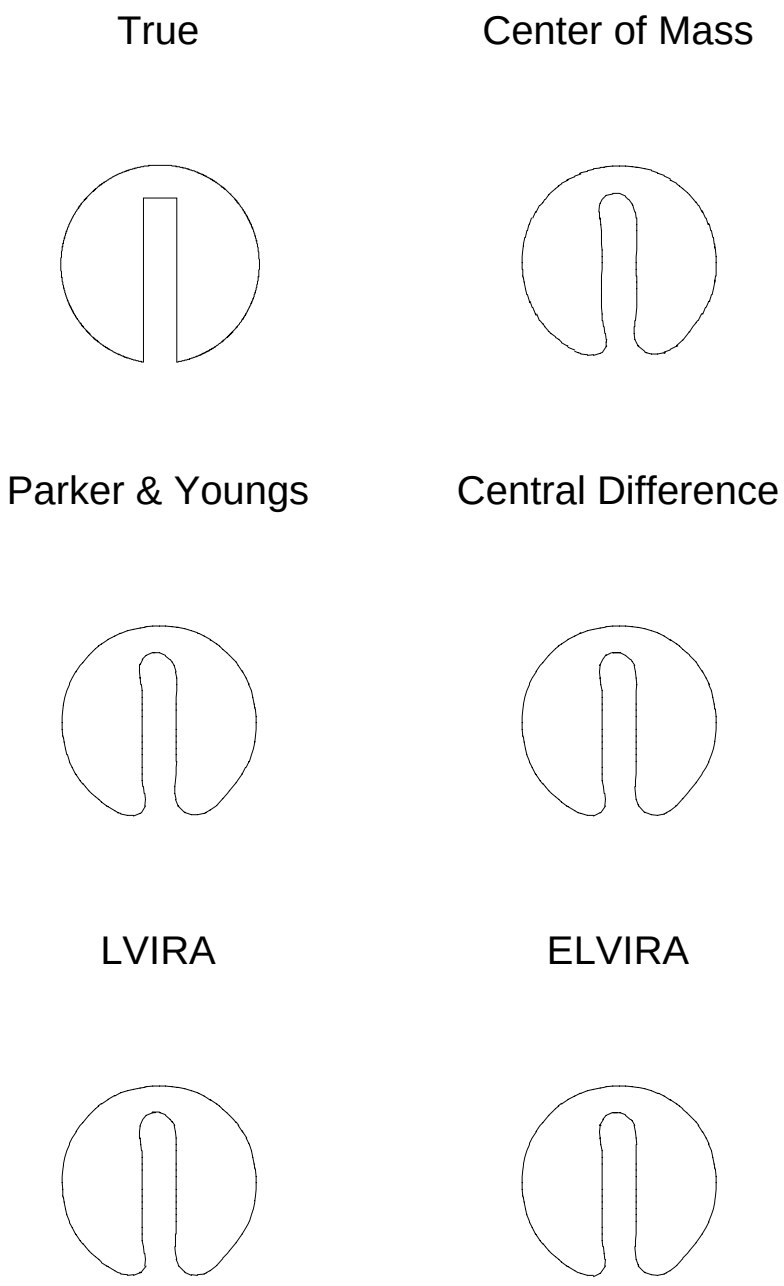


Figure 4.5 Here we present the results of using the unsplit advection algorithm and the various reconstruction methods to compute Zalesak's test problem. Note that the computations shown here and in Fig. 4.2 were conducted on the very coarse grid shown in Fig. 2.1. Consequently one cannot detect the increased resolution at the corners expected from the unsplit advection algorithm. Higher resolution computations of this problem do exhibit better resolution with unsplit algorithm.

	1/16	1/32	rate	1/64	rate
Center of Mass	7.0E-3	3.7E-3	0.95	2.2E-3	0.84
Parker & Youngs	5.8E-3	2.6E-3	1.12	1.3E-3	1.00
LVIRA	5.8E-3	2.7E-3	1.07	1.3E-3	1.04
ELVIRA	5.7E-3	2.6E-3	1.10	1.2E-3	1.08

Table 4.10 The average E^1 error for Zalesak's test problem.

	1/16	1/32	1/64
Center of Mass	8.9E-16	-1.9E-14	7.2E-14
Parker & Youngs	2.2E-15	1.6E-14	4.9E-14
LVIRA	-1.8E-15	-2.2E-15	3.3E-14
ELVIRA	-6.7E-15	-1.1E-14	1.0E-14

Table 4.11 The difference between the final and initial total area.

In Fig 4.5 we present the results of computing Zalesak's test problem on a grid with $h = 1/15$. Again, we chose this relatively coarse grid in order to facilitate a direct comparison with other published results of the same problem such as in [5] and [66]. The coarseness of the grid prevents one from detecting the increased resolution at the corners we expect with the unsplit advection algorithm. Higher resolution computations of this problem do exhibit better resolution at the corners of Zalesak's shape with unsplit algorithm. We conclude that, although both the fractional step and unsplit methods are second-order accurate, the unsplit method does indeed produce better overall results.

6. Concluding Remarks

We have presented a comprehensive framework for the design and implementation of modern volume-of-fluid interface tracking algorithms and conducted an extensive computational study of the accuracy of several commonly used versions of these algorithms. Our presentation is based on separating the interface reconstruction phase from the advection or time update phase of the overall tracking algorithm and studying the accuracy of the interface reconstruction algorithm independently of the advection algorithm. In our study of volume-of-fluid interface reconstruction algorithms, we have identified several key properties - or design criteria - that we believe will ensure that the method is second-order accurate on smooth interfaces; i.e., interfaces that have two or more continuous derivatives. In particular, we have found that if a volume-of-fluid interface reconstruction algorithm is designed in such a way that it always reproduces lines (or planes in 3D) exactly, then it will be second-order accurate on smooth interfaces in both the L^1 and the L^∞ norms when used to reconstruct stationary interfaces. We have introduced two new volume-of-fluid interface reconstruction algorithms that have this property and demonstrated that they consistently exhibit second-order accuracy when we use them to reconstruct smooth stationary interfaces, whereas the other algorithms we tested overall exhibit first-order accuracy.

In our study of volume-of-fluid advection algorithms we have demonstrated that one can obtain second-order accuracy (in space and time) by combining one of our second-order accurate interface reconstruction algorithms with a standard fractional step or operator split solution of the time evolution equation and alternating the sweep directions at each time step (i.e., Strang splitting). We have also introduced a new unsplit volume-of-fluid advection algorithm that is second-order accurate in space and time when combined with one of the second-order accurate interface reconstruction algorithms. Furthermore we have shown that the unsplit algorithm exhibits noticeably better resolution of regions near discontinuities in the derivatives of the interface (e.g., corners). Since this improved resolution does not manifest as an increase in the order of accuracy of the advection algorithm, we conjecture that it is a higher order effect due to an increase in the accuracy with which the algorithm resolves a portion of the error, such as the phase error (e.g., see the discussion on phase errors in [12]).

Another conclusion that can be drawn from our study is that piecewise linear interface reconstruction algorithms that reconstruct lines exactly will revert to first-order accuracy when the interface fails to be sufficiently smooth (e.g., remains continuous but has discontinuities in the first-derivative). We conjecture that the constraint that a volume-of-fluid interface reconstruction method must *always* reproduce the correct fluid volume in each cell is sufficient to guarantee first-order accuracy in the L^1 norm for time dependent advection problems - at least when the advection algorithm is formally second-order accurate as is the case in our studies. This conclusion appears to be true even for reconstruction algorithms that do not exhibit second-order accuracy on smooth interfaces, such as SLIC. However it is apparent from the results presented in Table 3.3 that something more than this constraint is needed in order to guarantee first-order accuracy in the L^∞ norm.

In summary, we have presented two new volume-of-fluid interface reconstruction algorithms and demonstrated that they are more accurate than the most commonly used volume-of-fluid interface reconstruction algorithms. These new interface reconstruction algorithms are currently being used in a number of application codes for modeling the motion of material interfaces in compressible gas dynamics [21, 48, 49, 32], high-pressure solids in the hydrostatic limit [35, 36, 50] and variable density incompressible fluid flow [3, 47]. We have also introduced a new, unsplit volume-of-fluid advection algorithm, demonstrated that it is second-order accurate in space and time and shown that it exhibits superior resolution of kinks or corners in the interface as compared to the fractional step advection algorithm, which is currently the most widely used advection algorithm.

Acknowledgements The authors would like to thank John Bell for many helpful discussions and especially for his suggestions concerning unsplit advection algorithms. The second author would also like to acknowledge Bill Noh's role introducing him to these problems and Phil Colella's role in encouraging him to pursue them.

References

[1] D. Adalsteinsson and J. A. Sethian. An overview of level set methods for etching, deposition, and lithography development. *IEEE Transactions On Semiconductor Manufacturing*, 10(1):167–184, February 1997.

[2] F. L. Addessio, D. E. Carroll, J. K. Dukowicz, F. H. Harlow, J. N. Johnson, B. A. Kashiwa, M. E. Maltrud, and H. E. Rupple. CAVEAT: A Computer Code for Fluid Dynamics Problems with Large Distortion and Internal Slip. Technical Report LA-10613-MS, Los Alamos National Laboratory, 1986.

[3] I. Aleinov and E. G. Puckett. Computing Surface Tension with High-Order Kernels. In *Proceedings of the 6th International Symposium on Computational Fluid Dynamics*, K. Oshima, editor, pages 6–13, Lake Tahoe, CA, 1995.

[4] N. Ashgriz and J. Y. Poo. FLAIR, Flux Line-Segment Model for Advection and Interface Reconstruction. *J. Comput. Phys.*, 93:449–468, 1991.

[5] J. B. Bell, C. N. Dawson, and G. R. Shubin. An Unsplit, Higher Order Godunov Method for Scalar Conservation Laws in Multiple Dimensions. *J. Comput. Phys.*, 74:1–24, 1988.

[6] A. R. Bishop, L. J. Campbell, and P. J. Channell. *Fronts, Interfaces and Patterns.* North-Holland, Amsterdam, 1984.

[7] C. Chan, J. Mazumder, and M. M. Chen. A Two-Dimensional Transient Model for Convection ina Laser Melted Pool. *Metallurgical Trans. A*, 15A:2175–2184, December 1984.

[8] A. J. Chorin. Flame Advection and Propagation Algorithms. *J. Comput. Phys.*, 35:1–11, 1980.

[9] P. Colella. Multidimensional Upwind Methods for Hyperbolic Conservation Laws. *J. Comput. Phys.*, 87:171–200, 1990.

[10] P. Colella, H. M. Glaz, and R. Ferguson. Multifluid Algorithms for Eulerian Finite Difference Methods. (unpublished manuscript), 1997.

[11] P. Colella, L. F. Henderson, and E. G. Puckett. A Numerical Study of Shock Wave Refraction at a Gas Interface. In *Proceedings of the AIAA 9th Computational Fluid Dynamics Conference*, number AIAA-89-1973, pages 426–439, Buffalo, New York, 1989.

[12] P. Colella and E. G. Puckett. *Modern Numerical Methods for Fluid Flow.* Cambridge University Press, 1997. (manuscript available via anonymous ftp on watt.berkeley.edu/e266).

[13] R. DeBar. A Method in Two-D Eulerian Hydrodynamics. Technical Report UCID-19683, Lawrence Livermore National Laboratory, March 1974.

[14] N. V. Deshpande. Fluid Mechanics of Bubble Growth and Collapse in a Thermal Ink-Jet Printhead. In *SPSE/SPIES Electronic Imaging Devices and Systems Symposium*, January 1989.

[15] A. F. Ghoniem, A. J. Chorin, and A. K. Oppenheim. Numerical Modeling of Turbulent Flow in a Combustion Tunnel. *Phil. Trans. R. Soc. Lond. A*, 304:303–325, 1982.

[16] J. Glimm and O. McBryan. A Computational Model for Interfaces. *Adv. Appl. Math.*, 6:422–435, 1985.

[17] J. J. Helmsen. *A Comparison of Three-Dimensional Photolithography Simulators.* PhD thesis, U. C. Berkeley, 1994.

[18] J. J. Helmsen, P. Colella, and E. G. Puckett. Non-Convex Profile Evolution in Two Dimensions Using Volume of Fluids. *J. Comput. Phys.*, January 1997. (submitted).

[19] J. J. Helmsen, P. Colella, E. G. Puckett, and M. Dorr. Two New Methods for Simulating Photolithography Development in Three Dimensions. In *Proceeedings of the 10th SPIE Optical/Laser Microlithography Conference*, volume 2726, pages 253–261, San Jose, CA, January 1996. SPIE.

[20] L. F. Henderson, P. Colella, and E. G. Puckett. On the Refraction of Shock Waves at a Slow-Fast Gas Interface. *J. Fluid Mech.*, 224:1–27, 1991.

[21] L. F. Henderson, E. G. Puckett, and P. Colella. Anomalous Refraction of Shock Waves. In *Shock Waves*, K. Takayama, editor, pages 283–286. Springer-Verlag, 1992.

[22] C. W. Hirt. *Flow-3D Users Manual*, 1988. Flow Sciences, Inc.

[23] C. W. Hirt and B. D. Nichols. Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries. *J. Comput. Phys.*, 39:201–225, 1981.

[24] K. S. Holian, S. J. Mosso, D. A. Mandell, and R. Henninger. MESA: A 3-D Computer Code for Armor/Anti-Armor Applications. Technical Report LA-UR–91-569, Los Alamos National Laboratory, 1991.

[25] D. B. Kothe, J. R. Baumgardner, S. T. Bennion, J. H. Cerutti, B. J. Daly, K. S. Holian, E. M. Kober, S. J. Mosso, J. W. Painter, R. D. Smith, and M. D. Torrey. PAGOSA: A Massively-Parallel, Multi-Material Hydro-Dynamics Model for Three-Dimensional High-Speed Flow and High-Rate Deformation. Technical Report LA-UR-92-4306, Los Alamos National Laboratory, 1992.

[26] D. B. Kothe and R. C. Mjolsness. RIPPLE: A New Model for Incompressible Flows with Free Surfaces. *AIAA Journal*, 30(11):2694–2700, November 1992.

[27] D. B. Kothe, R. C. Mjolsness, and M. D. Torrey. RIPPLE: A Computer Program for Incompressible Flows with Free Surfaces. Technical Report LA-12007-MS, Los Alamos National Laboratory, April 1991.

[28] B. LaFaurie, C. Nardone, R. Scardovelli, S. Zaleski, and G. Zanetti. Direct Numerical Simulation of Interface Breakup and Atomisation. In *Proceedings of ICLASS-94*, Rouen, France, July 1994.

[29] B. LaFaurie, C. Nardone, R. Scardovelli, S. Zaleski, and G. Zanetti. Modelling Merging and Fragmentation in Multiphase Flows with SURFER. *J. Comput. Phys.*, 113:134–147, 1994.

[30] R. J. LeVeque. High-Resolution Conservative Algorithms for Advection in Incompressible Flow. *SIAM J. Numer. Anal.*, 33(2):627–665, April 1996.

[31] H. Liu, E. J. Lavernia, and R. H. Rangel. Numerical Investigation of Micropore Formation During Substrate Impact of Molten Droplets in Plasma Spray Processes. *Atomization and Sprays*, 4:369–384, 1994.

[32] D. L. Marcus, E. G. Puckett, J. B. Bell, and J. S. Saltzman. Numerical Simulation of Accelerated Interfaces. In *3rd International Workshop on the Physics of Compressible Turbulent Mixing*, R. Dautray, editor, pages 63–81. CEA DAM, 1991.

[33] J. M. McGlaun, S. L. Thompson, C. N. Kmetyk, and M. G. Elrick. A Brief Description of the Three Dimensional Shock Wave Physics Code CTH. Technical report, Sandia National Laboratory, 1992.

[34] H. J. Melosh and M. E. Kipp. Giant Impact Theory of the Moon's Origin: First 3-D Hydrocode Results. In *Lunar and Planetary Science XX*. Lunar and Planetary Institute, Houston, 1989. (abstract).

[35] G. H. Miller and E. G. Puckett. Edge Effects in Molybdenum-Encapsulated Molten Silicate Shock Wave Targets. *J. Appl. Phys.*, 73(3):1426–1434, February 1994.

[36] G. H. Miller and E. G. Puckett. A High-Order Godunov Method for Multiple Condensed Phases. *J. Comput. Phys.*, 128:134–164, August 1996.

[37] B. D. Nichols, C. W. Hirt, and R. S. Hotchkiss. SOLA-VOF: A Solution Algorithm for Transient Fluid Flow with Multiple Free Boundaries. Technical Report LA-8355, Los Alamos National Laboratory, August 1980.

[38] W. F. Noh and P. R. Woodward. SLIC (Simple Line Interface Calculation). In *Lecture Notes in Physics; 59*, A. I. van der Vooren and P. J. Zandbergen, editors, pages 330–340. Springer-Verlag, 1976.

[39] H. N. Oguz and A. Prosperetti. Dynamics of Bubble Growth and Detachment from a Needle. *J. Fluid Mech.*, 257:111–145, Dec 1993.

[40] S. Osher and J. A. Sethian. Fronts Propagating with Curvature-Dependent Speed: Algorithms based on Hamilton-Jacobi Formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[41] B. J. Parker and D. L. Youngs. Two and Three Dimensional Eulerian Simulation of Fluid Flow with Material Interfaces. Technical Report 01/92, UK Atomic Weapons Establishment, Aldermaston, Berkshire, Feb 1992.

[42] J. E. Pilliod. An Analysis of Piecewise Linear Interface Reconstruction Algorithms for Volume-Of-Fluid Methods. Master's thesis, U. C. Davis, September 1992.

[43] J. E. Pilliod. *A Second-Order Unsplit Method for Modeling Flames in Two-Dimensional Compressible Flow*. PhD thesis, University of California, Davis, September 1996.

[44] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.

[45] E. G. Puckett. A Numerical Study of Shock Wave Refraction at a CO_2/CH_4 Interface. In *Multidimensional Hyperbolic Problems and Computations*, J. Glimm and A. J. Majda, editors, volume 29 of *IMA Volumes in Mathematics and Its Applications*, pages 261–280. Springer-Verlag, 1991.

[46] E. G. Puckett. A Volume-of-Fluid Interface Tracking Algorithm with Applications to Computing Shock Wave Refraction. In *Proceedings of the 4th International Symposium on Computational Fluid Dynamics*, H. Dwyer, editor, pages 933–938, Davis, CA, 1991.

[47] E. G. Puckett, A. S. Almgren, J. B. Bell, D. L. Marcus, and W. J. Rider. A High-Order Projection Method for Tracking Fluid Interfaces in Variable Density Incompressible Flows. *J. Comput. Phys.*, 130:269–282, 1997.

[48] E. G. Puckett and L. F. Henderson. The Anomalous Refraction of Shock Waves in Gases. *Proc. Royal Soc. London A*, 1997. (submitted).

[49] E. G. Puckett, L. F. Henderson, and P. Colella. A General Theory of Anomalous Refraction. In *Shock Waves @ Marseilles*, R. Brun and L. Z. Dumitrescu, editors, volume IV, pages 139–144. Springer-Verlag, 1995.

[50] E. G. Puckett and G. H. Miller. The Numerical Computation of Jetting Impacts. In *Proceedings of the 20th International Symposium on Shock Waves*, B. Sturtevant, J. Sheperd, and H. Hornung, editors, volume II, pages 1467–1472, California Institute of Texhnology, July 1996. World Scientific.

[51] E. G. Puckett and J. S. Saltzman. A 3-D Adaptive Mesh Refinement Algorithm for Multimaterial Gas Dynamics. *Physica D*, 60:84–104, 1992.

[52] W. J. Rider and D. B. Kothe. Reconstructing Volume Tracking. *J. Comput. Phys.*, page (submitted), 1997.

[53] J. A. Sethian. Turbulent Combustion in Open and Closed Vessels. *J. Comput. Phys.*, 54:425–456, 1984.

[54] J. A. Sethian. Tracking interfaces with level sets. *AMERICAN SCIENTIST*, 85(3):254–263, May-June 1997.

[55] J. Strain. A Boundary Integral Approach to Unstable Solidification. *J. Comput. Phys.*, 85:342–389, 1989.

[56] M. Sussman, P. Smereka, and S. Osher. A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow. *J. Comput. Phys.*, 114(1), 1994.

[57] P. A. Torpey. Prevention of Air Ingestion in a Thermal Ink-Jet Device. In *Proceedings of the 4th International Congress on Advances in Non-Impact Print Technologies*, March 1988.

[58] M. D. Torrey, L. D. Cloutman, R. C. Mjolsness, and C. W. Hirt. NASA-VOF2D: A Computer Program for Incompressible Flows with Free Surfaces. Technical Report LA-10612-MS, Los Alamos National Laboratory, December 1985.

[59] M. D. Torrey, R. C. Mjolsness, and L. R. Stein. NASA-VOF3D: A Three-Dimensonal Computer Program for Incompressible Flows with Free Surfaces. Technical Report LA-11009-MS, Los Alamos National Laboratory, July 1987.

[60] G. Trapaga, E. F. Matthys, J. J. Valencia, and J. Szekely. Fluid Flow, Heat Transfer, and Solidificaton of Molten Metal Droplets Impinging on Substrates - Comparison of Numerical and Experimental Results. *Metall. Trans. B.*, 23(6):701–718, 1992.

[61] S. O. Unverdi and G. Tryggvason. Computations of Multi-Fluid Flows. *Physica D*, 60:70–83, 1992.

[62] B. van Leer. Towards the Ultimate Conservative Difference Scheme. V. A Second-Order Sequel to Godunov's Method. *J. Comput. Phys.*, 32:101–136, 1979.

[63] D. B. Wallace. A Method of Characteristics Model of a Drop-on-Demand Ink-Jet Device Using an Integral Method Drop Formation Model. In *Proceedings of the ASME Winter Annual Meeting*, number 89-WA/FE-4, December 10–15 1989. San Francisco, CA.

[64] N. Whitaker. Numerical Solution of the Hele-Shaw Equations. *J. Comput. Phys.*, 90:176–199, 1990.

[65] D. L. Youngs. Time-Dependent Multi-material Flow with Large Fluid Distortion. In *Numerical Methods for Fluid Dynamics*, K. W. Morton and M. J. Baines, editors. Academic Press, 1982.

[66] S. T. Zalesak. Fully Multidimensional Flux-Corrected Transport Algorithms for Fluids. *J. Comput. Phys.*, 31:335–362, 1979.

[67] J. Zhu and J. A. Sethian. Projection Methods Coupled to Level Set Interface Techniques. *J. Comput. Phys.*, 102:128–138, 1992.